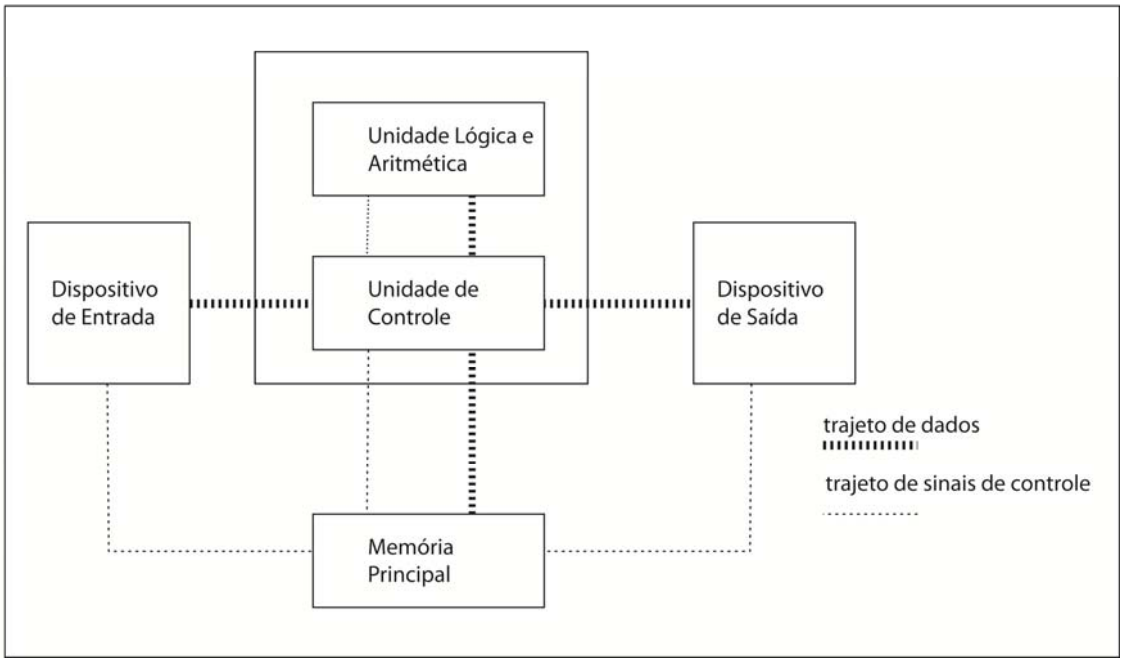
Programação é o processo de escrita, teste e manutenção de um programa de computador. O programa é escrito numa linguagem de programação.



Dispositivo de entrada (o teclado)	Introdução de dados;
Unidade Lógica e Aritmética (ULA)	Parte responsável pelas operações matemáticas e avaliações lógicas;
Unidade de Controlo	Exerce controlo sobre as demais partes do nosso computador. É quem distribui tarefas às outras unidades;
Memória (RAM)	Guarda o algoritmo a ser executado e os dados a serem utilizados pelo mesmo (os dados fornecidos ao computador e o resultado das suas operações ficam guardados na memória);
Dispositivo de Saída (monitor e impressora)	Apresentação dos resultados obtidos;

Resumidamente, podemos afirmar que existem quatro operações básicas que qualquer computador pode executar:

a) operações de entrada e saída: ler dados do teclado e escrever dados no monitor são exemplos destas operações. Servem para introduzir dados na memória e mostrar dados que lá estejam armazenados;

b) operações aritméticas: são utilizadas na realização de operações matemáticas (adição, subtração, multiplicação e divisão);

c) operações lógicas e relacionais: comparações, testes de condições lógicas (2>6 ? X=Y ?);

d) movimentação de dados entre os vários componentes: as operações aritméticas são executadas na Unidade Lógica e Aritmética, necessitando da transferência dos dados para essa unidade e da volta do resultado final para ser guardado na memória.

Exemplo: resolvendo um problema

Suponha que queríamos resolver o seguinte problema: dados dois números, calcular a sua soma. Uma possível solução seria a seguinte:

a) saber quais são os números;
b) calcular a soma dos números;
c) responder à questão com o valor do resultado.

Resolução em termos das operações básicas citadas anteriormente:

a) operação de entrada de dados dos números;
b) Unidade Lógica e Aritmética (ULA)
    a. movimento do valor dos números entre a memória e a ULA;
    b. operação aritmética de somar os 2 números;
    c. movimentação do resultado da ULA para guardar na memória;
c) operação de saída do resultado, que está guardado na memória

Em resumo, pode-se dizer que escrever algoritmos ou consiste em dividir qualquer problema em muitos pequenos passos, usando uma ou mais das quatro operações básicas citadas.

Esses passos que compõem o algoritmo são denominados de comandos. Os comandos de uma linguagem de programação podem estar mais próximos da máquina (linguagens de baixo nível) ou serem mais facilmente entendidos pelo homem (linguagens de alto nível).

Uma **linguagem de programação** é um método padronizado para expressar instruções para um computador. É um conjunto de regras sintácticas e semânticas usadas para definir um programa de computador. Uma linguagem permite que um programador especifique precisamente sobre quais dados um computador vai actuar, como estes dados serão armazenados ou transmitidos e quais acções devem ser tomadas sob várias circunstâncias. O conjunto de palavras, compostos de acordo com essas regras, constitui o código fonte[1] de um *software*[2]. Esse código fonte é depois traduzido para código de máquina[3], que é executado pelo processador.

[1] Código fonte ou *source code* em inglês, é o conjunto de palavras escritas de forma ordenada, contendo instruções numa das linguagens de programação existentes no mercado, de maneira lógica

Um **programa de computador** é essencialmente um algoritmo que diz ao computador os passos específicos e em que ordens devem ser executados, como por exemplo, os passos a serem tomados para calcular as notas que serão impressas nos boletins dos alunos de uma escola.

Um **algoritmo** é uma sequência não ambígua de instruções que é executada até que determinada condição se verifique.

Do livro "Linguagens de Programação" para o Ensino Profissional, Artur Augusto Azul, Porto Editora:

- Um algoritmo é uma sequência finita e logicamente ordenada de acções para se chegar à solução de um problema.
- Um algoritmo correctamente formulado permite-nos chegar sempre aos mesmos resultados, desde que os dados ou condições de partida sejam os mesmos.

O conceito de algoritmo é frequentemente ilustrado pelo exemplo de uma receita. Eles podem repetir passos ou necessitar de decisões até que a tarefa seja finalizada.

O termo receita está sendo empregue com o sentido de "lista de actividades que leva ao fim desejado". Isto é, um algoritmo é uma lista de actividades que, se executadas coerentemente, conforme estipulado na receita, leva sempre à resposta desejada.

Exemplo 1
Substituir uma lâmpada fundida

1. Preparar uma lâmpada nova
2. Retirar a lâmpada fundida
3. Colocar a lâmpada nova
4. Ligar o interruptor

Exemplo 2
Substituir um pneu furado

1. Preparar o pneu sobressalente
2. Colocar o macaco na posição adequada
3. Levantar o carro
4. Retirar o pneu furado
5. Colocar o pneu sobressalente

Exemplo 3

---

[2] *Software* ou programa de computador é uma sequência de instruções a serem seguidas e/ ou executadas, na manipulação, redireccionamento ou modificação de um dado/ informação ou acontecimento

[3] todo os computadores possuem um conjunto de instruções que o seu processador é capaz de executar. Essas instruções, chamadas código máquina, são representadas por sequências de bits

Preparar bifes à milaneza

1. Limpar a peça de carne
2. Cortar a carne em bifes
3. Colocar farinha num prato
4. Juntar 2 ovos e mexer
5. Repetir, para cada bife
   5.1. Passar o bife na mistura de farinha, nos 2 lados
   5.2. Levar bife à frigideira
   5.3. Esperar que fique douradinho
   5.4. Virar o bife
   5.5. Esperar que fique douradinho
   5.6. Retirar bife e colocar sobre uma toalha de papel até secar
6. Retirar do papel toalha e colocar numa travessa
7. Decorar a travessa com folhas de alface
8. Servir

Exemplo 4
Calcular a área de um rectângulo

1. Obter os valores do comprimento
2. Calcular área = comprimento x largura
3. Apresentar o valor

Conceito de **constante**

Uma constante é um determinado valor fixo que não se modifica ao longo do tempo, durante a execução de um algoritmo. Por exemplo pi = 3.1415

A constante é classificada como sendo numérica (números), lógica (verdadeiro ou falso, um ou zero) ou literal (qualquer conjunto de caracteres).

Numérica: a representação de uma constante numérica é feita no sistema decimal, podendo ser um número com ou sem parte fraccionária. Pode ser negativa ou positiva de acordo com o sinal que precede os algarismos que compões o número.

a) 15
b) -15

Lógica: Pode assumir os valores Verdadeiro ou Falso, Um ou Zero.

Literal: Qualquer sequência de caracteres. Deverá ser colocada entre aspas para que não seja confundida com outro elemento qualquer.

a) ‘Mensagem’
b) “1234”
c) “Rui Silva”

Conceito de **Variável**

Sabe-se da matemática que uma variável é uma representação simbólica dos elementos de um certo conjunto. Nos algoritmos a cada variável corresponde uma zona na memória, cujo conteúdo pode variar ao longo do tempo durante a execução do programa.

Embora uma variável possa assumir diferentes valores, **só pode armazenar um único valor em cada instante**.

A variável é **identificada por um nome**, formado por um ou mais caracteres devendo o primeiro ser uma letra. Alguns símbolos não podem ser utilizados para formar o nome de variáveis por se encontrarem reservados:

Por exemplo:
/ * + - , ;

Também não é possível utilizar para definir o nome de uma variável as instruções utilizadas na linguagem, por exemplo

LER, ESCREVER, etc.

Exemplo de nomes permitidos:

a X5 Nota_Final LadoQuadrado

Exemplo de nomes não permitidos:

123a X5* Nota Final LadoQuadrado-

É recomendável que os nomes sejam os mais significativos possíveis, isto é, que designem da melhor maneira a natureza dos valores que nelas estão armazenados.

Declaração de Variáveis

As variáveis só podem armazenar valores de um mesmo tipo. Conforme o seu tipo a variável é classificada como sendo numérica, lógica ou literal.

Para indicar o tipo de uma variável é usada a declaração de variáveis. No momento em que se declara uma variável, é feita a associação do nome escolhido com a respectiva zona de memória que o mesmo passa a simbolizar. Qualquer referência ao nome implica a referência ao conteúdo da zona de memória.

A declaração de variáveis obedece à seguinte sintaxe:

DECLARAR lista_de_nomes_das_variáveis nome_do_tipo-de_dados

Exemplo:

```
DECLARAR nota, codigo, soma NUMÉRICO
DECLARAR ok, option LÓGICO
DECLARAR nome, aluno_morada LITERAL
```

Comentários

O comentário é um instrumento muitas vezes ignorado mas de grande valia para a compreensão do algoritmo. Consiste num texto que aparece delimitado por um caracter especial (chavetas, pelicas, dupla barra). O comentário é ignorado e serve unicamente para explicar o significado do código onde foi inserido.

Exemplo:

```
DECLARAR codigo NUMÉRICO          ' guarda o valor do código do aluno
```

Expressões

Uma expressão é um conjunto de caracteres que pode representar variáveis ou constantes separados por operadores. As expressões podem ser aritméticas, literais, relacionais ou lógicas.

$$\left[\frac{2}{3}+(5-3)\right]+1=$$

tradicional

(2/3+(5-3))+1=

computacional

Expressões **aritméticas**

Denomina-se expressão aritmética aquela cujos operadores são aritméticos e cujos operandos são constantes e/ ou variáveis do tipo numérico.

O conjunto de operações básicas adoptado é o que se conhece da matemática:

Adição	+
Multiplicação	*
Potenciação	^

Subtracção	-
Divisão	/

As operações obedecem às seguintes regras de prioridade:

1º Potenciação e Radiciação
2º Multiplicação e Divisão
3º Adição e Subtracção

A utilização de parêntesis pode alterar a ordem de prioridades.

Expressões **literais**

Uma expressão literal é formada por operadores literais e operandos que são constantes e/ ou variáveis do tipo literal.

Existem várias operações possíveis com literais. O conjunto de operações básico adoptado varia com a linguagem, neste módulo vamos usar é o seguinte:

Concatenação	+

Exemplo: supondo que a variável PrimeiroNome contém o literal “Rui “ e a variável UltimoNome contém o literal “Silva“, o valor fornecido pela expressão PrimeiroNome + UltimoNome seria o literal “Rui Silva”.

Expressões **Relacionais**

Uma expressão relacional, ou simplesmente uma relação, é uma comparação realizada entre dois valores do mesmo tipo básico. Estes valores são representados na relação através de constantes, variáveis e/ ou expressões. Os operadores relacionais que indicam a comparação a ser realizada entre dois termos do mesmo tipo básico são:

Igual a	=
Diferente de	!= ou <>
Menor que	<
Maior que	>
Menor ou igual a	<=
Maior ou igual a	>=

O resultado obtido de uma relação é **sempre um valor lógico.**

Expressões **Lógicas**

É comum nos algoritmos surgirem situações em que a execução de uma acção ou uma sequência de acções está sujeita a condições. Esta condição é representada no texto do algoritmo por meio de uma expressão lógica.

Denomina-se expressão lógica à expressão cujos operadores são lógicos e cujos operadores são relações, constantes e/ ou variáveis do tipo lógico. Os operadores lógicos são:

Conjunção	E
Disjunção	OU
Negação	NÃO

As tabelas de verdade destes operadores são as seguintes:

Tabela da Conjunção (E)				
A		B		A E B
Verdadeiro	E	Verdadeiro	=	Verdadeiro
Verdadeiro	E	Falso	=	Falso

Verdadeiro	E	Verdadeiro	=	Falso
Verdadeiro	E	Falso	=	Falso

Tabela da Disjunção (OU)				
A		B		A OU B
Verdadeiro	OU	Verdadeiro	=	Verdadeiro
Verdadeiro	OU	Falso	=	Verdadeiro
Falso	OU	Verdadeiro	=	Verdadeiro
Falso	OU	Falso	=	Falso

Tabela da Negação (NÃO)	
NÃO Verdadeiro	Falso
NÃO Falso	Verdadeiro

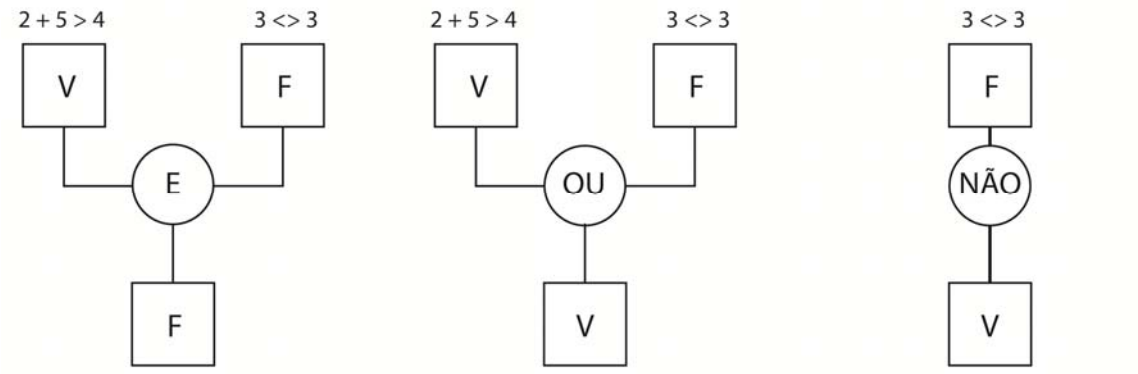
As operações obedecem à seguinte regra de prioridade.

1º NÃO
2º E
3º OU

Também nas operações lógicas vários níveis de parêntesis podem ser utilizados com a finalidade de estabelecer uma nova ordem de execução entre operadores lógicos.

O resultado obtido da avaliação de uma expressão lógica é sempre um valor lógico, isto é True ou False, Verdadeiro ou Falso, 1 ou 0.

Exemplo:



Atribuição de valores

O processo de atribuição de valores permite colocar valores de qualquer tipo básico numa variável previamente definida.

A sintaxe utilizada para atribuir um determinado valor a uma variável é a seguinte:

Nome_da_variavel = expressão_do_tipo_da_variavel ou valor

Nome_da_variavel = Nome_da_variavel_a_ser_atribuida

```
Algoritmo
    Declarar Largura, Comprimento, Area Numerico
    Largura = 75
    Comprimento = 100
    Area = Largura * Comprimento
Fim de Algoritmo
```

Funções

Além das operações aritméticas, anteriormente citadas, podem usar-se nas expressões aritméticas algumas funções muito comuns na matemática.

Abs(...) devolve o valor absoluto do argumento

Exemplo:

```
Y = - 125
Valor =  Abs(Y)
```
O conteúdo de Valor é igual a 125

Int(...) devolve a parte inteira do argumento

Exemplo:

```
P = 123.75
Q =  Int(P)
```
O conteúdo de Q é igual a 123

Sqr(...) devolve o quadrado do argumento

Exemplo:

```
H = 12
Quadrado =  Sqr(H)
```
O conteúdo de Quadrado é igual a 144

Sqrt(...) devolve a raíz quadrada do argumento

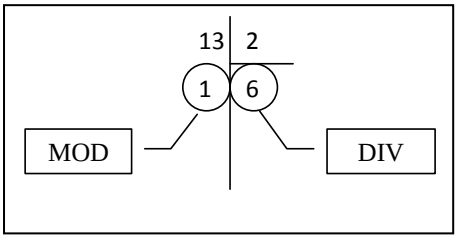
Exemplo:

```
Numero = 81
Raiz =  Sqrt(Numero)
```
O conteúdo de Raiz é igual a 9

Div Devolve o resultado da divisão inteira

Exemplo:

```
q = 13
resultado =  q Div 2
```
O conteúdo de q é igual a 6



Mod Devolve o resto da divisão

Exemplo:

m = 13
resultado = m Div 2
O conteúdo de q é igual a 1

São ainda permitidas as funções trigonométricas.

Comandos de Entrada e Saída

Seja a seguinte situação: início de um programa que se encontra armazenado na memória principal de um computador. Como e quem determina o momento de entrada e/ ou saída de dados para o programa e a saída de resultados para o utilizador?

Isto é tarefa do programador quando descreve as acções a serem executadas. Os comandos de entrada e saída são as ferramentas para esta finalidade.

A sintaxe do comando de entrada é a seguinte:

LER lista_de_identificadores

Em que LER é uma palavra reservada e lista_de_identificadores representa o nome das variáveis e/ ou constantes, separadas por vírgulas, nas quais serão armazenados os valores provenientes de um meio de entrada (por exemplo o teclado).

Exemplo:

```
DECLARAR   codigo, nota  NUMÉRICO

...

LER codigo, nota
```

O programa através do comando (ou instrução) LER, espera que sejam introduzidos dois valores numéricos a partir de uma unidade de entrada. Os valores serão armazenados nas zonas de memória identificados pelos nomes *codigo* e *nota*.

A sintaxe do comando de saída é a seguinte:

ESCREVER lista_de_identificadores

Em que ESCREVER é uma palavra reservada e lista_de_identificadores representa o nome das variáveis e/ ou constantes, separadas por vírgulas, cujos conteúdos serão mostrados ao utilizador através de um meio de saída (normalmente o monitor).

Exemplo:

```
DECLARAR   idade  NUMÉRICO
DECLARAR   nome  LITERAL
```

...

idade = 18
nome = “Rui Silva”
ESCREVER idade
ESCREVER nome

ou de uma forma mais elaborada

ESCREVER “O ”, nome, “ tem ”, idade, “ anos.”

O conteúdo das variáveis *nome* e *idade* será exibido através de uma unidade de saída.

```
Algoritmo
    Declarar Largura, Comprimento, Area Numerico
    Escrever “Insira o valor da largura do terreno:”
    Ler Largura
    Escrever “Insira o valor do comprimento do terreno:”
    Ler Comprimento
    Area = Largura * Comprimento
    Escrever Area
Fim de Algoritmo
```

Exercícios

Escreva um algoritmo que:

a) Leia um valor e escreva esse valor ao cubo.
b) Leia um valor e escreva o simétrico do valor lido
c) Leia um valor referente ao raio de uma circunferência
    a. Escreva a área do círculo
    b. O perímetro da circunferência
d) Leia dois valores numéricos e escreva a sua soma.
e) Leia três valores: *a*, *b* e *c* calcule as raízes da função $ax^2 + bx + c$.
f) Leia dois valores para duas variáveis de nome PrimeiroValor e SegundoValor, escreva o conteúdo das variáveis que acabou de ler. Troque os valores das variáveis e escreva o resultado.

Estruturas básicas

Estrutura **sequencial**

Num algoritmo aparecem em primeiro lugar as declarações de variáveis seguidas pelos comandos ou instruções que, se não houver indicação em contrário, deverão ser executados numa sequência linear.

```
Algoritmo
    d1
    d2
    dn
    ...
```

em que:

d1, d2, ..., dn são declarações

```
    c1
    c2                  c1, c2, ..., cn são comandos
    cn
Fim de Algoritmo
```

Exemplo de um algoritmo:

a) Dadas duas variáveis a e b, colocar numa terceira variável c, o resultado de (a + b) x b.

```
Algoritmo
    Declarar a, b, c numérico
    Ler a, b
    c = (a + b) * b
    Escrever c
Fim de algoritmo
```

b) Dadas duas variáveis a e b numéricas, trocar os seus conteúdos e escrever a e b.

```
Algoritmo
    Declarar a, b numérico
    Ler a, b
    Escrever a, b
    b = a + b
    a = b - a
    b = b - a
    Escrever a, b
Fim de algoritmo
```

Estrutura **Condicional**

A estrutura condicional permite a escolha do grupo de acções e estruturas a serem executadas quando determinada condição, representada por uma expressão lógica, é ou não satisfeita.

Estrutura condicional Simples:

```
SE condição ENTÃO
    sequência de    comandos
FIM DE SE
```

A sequência de comandos só será executada se a condição for verdadeira.

Exemplo:
Ler três valores numéricos escrever uma mensagem se a soma dos dois primeiros números for maior do que o terceiro número.

```
Algoritmo
    Declarar a, b, c numérico
    Ler a, b, c
```

```
    Se a + b > c então
        Escrever "Mensagem"
    Fim de se
Fim de algoritmo
```

Estrutura condicional Composta:

```
SE condição ENTÃO
    sequência de comandos A
SENÃO
    sequência de comandos B
FIM DE SE
```

A sequência de comandos A só será executada se a condição for verdadeira, a sequência de comandos B será executada se a condição for falsa.

Exemplo:
Dadas duas variáveis a e b numéricas, verificar se a é igual a b. Se a condição for verdadeira, colocar o valor 1 na variável x e o valor 2 na variável y. Se for falsa colocar o valor de x igual a -2 e o valor de y igual a -1. No final escrever x e y.

```
Algoritmo
    Declarar a, b, x, y numérico
    Ler a, b
    Se a = b então
        x = 1
        y = 2
    Senão
        x = -2
        y = -1
    Fim de se
    Escrever x, y
Fim de algoritmo
```

Exercícios

Escreva um algoritmo que:

a) Leia dois valores numéricos e escreva o maior deles.
b) Leia três valores numéricos e escreva o maior deles – preveja a situação de serem todos iguais.
c) Leia dois valores numéricos. Se a sua soma for superior a dez escreva a mensagem "soma maior do que 10", senão escreva a mensagem "soma menor que dez".

Estrutura **de repetição**

A estrutura de repetição permite que uma sequência de comandos seja executada repetidamente até que uma determinada condição de interrupção seja satisfeita (a condição de interrupção é representada por uma questão lógica).

Esta estrutura pode apresentar-se das seguintes formas:

Estrutura **REPETIR**

Interrupção no início:

```
REPETIR

    SE condição ENTÃO
        INTERROMPER
    FIM DE SE

    Sequência de comandos

FIM DE REPETIR
```

Nesta estrutura, a sequência de comandos será repetida enquanto a condição não for satisfeita. Quando a condição se verificar, a repetição será interrompida e a sequência de comandos que vier logo após a expressão fim de repetir será executada.

Exemplo:

Calcular a soma dos números entre 100 e 200, inclusivé. No final escrever o total.

```
Algoritmo
    Declarar Numero, Soma Numérico
    Soma =  0
    Numero =   100

    Repetir
        Se Numero > 200 então
            Interromper
        Fim de se
        Soma = Soma + Numero
        Numero = Numero +1
    Fim de repetir

    Escrever Soma
Fim de algoritmo
```

Interrupção no interior

```
REPETIR
    Sequência de comandos A

    SE condição ENTÃO
        INTERROMPER
    FIM DE SE

    Sequência de comandos B
FIM DE REPETIR
```

Nesta estrutura, as sequências comandos A e B, serão repetidas até que a condição seja satisfeita. Quando isto ocorrer a repetição será interrompida e a sequência de comandos que vier logo após a expressão *fim de repetir* será executada.

Exemplo:

Calcular a soma dos números entre 100 e 200, inclusivé. No final escrever o total.

```
Algoritmo
    Declarar Numero, Soma Numérico
    Soma =  0
    Numero =   99
    Repetir
        Numero = Numero +1
        Se numero > 200 então
            Interromper
        Fim de se
        Soma = Soma + Numero
    Fim de repetir
    Escrever Numero
Fim de algoritmo
```

Interrupção no fim

```
REPETIR

    Sequência de comandos

    SE condição ENTÃO
        INTERROMPER
    FIM DE SE

FIM DE REPETIR
```

Nesta estrutura, a sequência de comandos será repetida até que a condição seja satisfeita. Quando isto acontecer, a repetição será interrompida e a sequência de comandos que vier logo após a expressão fim de repetir será executada.

Exemplo:

Calcular a soma dos números entre 100 e 200, inclusivé. No final escrever o total.

```
Algoritmo
    Declarar Numero, Soma Numérico
    Soma =  0
    Numero =   99
    Repetir
        Numero = Numero +1
        Soma = Soma + Numero
        Se numero = 200 então
            Interromper
        Fim de se
    Fim de repetir
    Escrever Numero
Fim de algoritmo
```

Estrutura **PARA**

```
PARA Variável_Controlo = ValorInicial ATÉ ValorFinal FAZER
    Sequência de comandos
FIM DE PARA
```

Variável_Controlo é uma variável inteira, que será incrementada ou decrementada a partir do valor inicial até que atinja o valor final previamente estabelecido.

Valor_Inicial é o primeiro valor que a variável de controlo irá assumir. Valor_Final é o valor limite que a variável de controlo poderá assumir.

O PARA é uma estrutura de repetição incondicional, isto é, uma vez iniciada será executada quantas vezes estiver estipulado no intervalo entre ValorInicial e ValorFinal.

Exemplo:

```
Algoritmo
    Declarar I, A, B Numérico

    Escrever("Menor número: ")
    Ler A
    Escrever("Maior número: ")
    Ler B

    Para I = A até B fazer
        Escrever I
    Fim Para

Fim de Algoritmo
```

Variáveis Compostas

Variáveis Compostas Unidimensionais, ou vectores correspondem a posições de memória identificadas pelo mesmo nome e individualizadas por um índice, cujo conteúdo é do mesmo tipo.

A atribuição de valores a cada posição de um vector processa-se da mesma forma como nas outas variáveis, com a excepção de ser necessário indicar especificamente qual a posição onde se irá efectuar essa atribuição.

Por exemplo:

```
Algoritmo
    Declarar numeros[5] numérico
    numeros[0] = 35
    numeros[1] = 21
    numeros[2] = 19
    numeros[3] = 5
    numeros[4] = 15
    numeros[5] = 43
    ....
Fim de algoritmo
```

Graficamente poderíamos representar o vector da seguinte forma:

Modularização

Subprogramas (rotinas) são grupos de instruções de um programa.

Permitem a divisão de um programa em vários módulos, cada um dos quais pode ser desenvolvido separadamente.

Uma rotina pode executar as mesmas acções que um programa (receber valores, manipulá-los e/ou produzir novos valores) contudo é geralmente utilizada para fins específicos.

Exemplo:

Escreva um algoritmo que calcule a área de um terreno rectangular. Utilize um Subprograma para efectuar o cálculo.

```
Algoritmo
    Declarar Lado1, Lado2 Numérico

    Subprograma CalculaArea
        Escrever Lado1 * Lado2
    Fim de Subprograma

    Ler Lado1, Lado2
    CalculaArea

Fim de algoritmo
```

Variáveis locais e variáveis globais

O domínio de uma variável define-se como sendo a gama de instruções na qual a variável é conhecida e a sua vida define-se como o intervalo de tempo durante o qual determinada porção de memória do computador está associado à variável.

Diz-se que determinadas variáveis são **locais**, quando apenas podem ser usadas localmente – dentro de um subprograma.

Todas as variáveis utilizadas num dado subprograma que não tenham sido declaradas nesse subprograma, são denominadas **variáveis não locais** e podem ser divididas em dois grupos:

- Aquelas que são definidas no programa principal, e são portanto utilizadas por todos os subprogramas do programa, designam-se por **variáveis globais**.
- Aquelas que embora não sejam globais, são utilizadas pelos subprogramas, às quais se dá o nome de **variáveis livres**.

Exemplo:

Escreva um algoritmo que calcule a área de um terreno circular. Utilize um Subprograma para efectuar o cálculo.

```
Algoritmo
    Declarar Raio Numérico              ' Raio é uma variável global

    Subprograma CalculaArea
        Declarar Pi, Area Numérico      ' Pi e Area são variáveis locais, o seu ciclo
        Pi = 3.1415                     de vida é o intervalo de tempo durante o
        Area = Pi * Sqr(Raio)           qual determinada porção de memória do
        Escrever Area                   computador está associado à variável
    Fim de Subprograma

    Ler Raio
    CalculaArea                         ' Chamada do subprograma

Fim de algoritmo
```

Projectar um algoritmo

Quando se elabora um algoritmo para um problema simples, uma solução pode ocorrer repentinamente depois de se pensar no problema como um todo.

Contudo, projectar aplicações práticas para o computador não é assim tão simples, normalmente programas deste género têm milhares de instruções.

Este tipo de trabalho é tão complexo como projectar uma máquina com milhares de peças. Para que tudo funcione como projectado, todas as peças devem ajustar-se nos seus devidos lugares constituindo uma unidade organizada e operacional.

Projectar um algoritmo é um procedimento semelhante a escrever um livro, começa-se com uma ideia geral do que se deseja abordar. Prossegue-se com a selecção dos temas desejados, a organização dos capítulos, a recolha do material bibliográfico e a escolha das palavras apropriadas para transmitir o correcto significado ao leitor. Quando se projecta um algoritmo também se começa com uma ideia geral do que se pretende fazer. Deve-se então organizar as ideias e escolher a sequência correcta de instruções para que o computador execute as acções desejadas.

Um bom método para começar é fazer um esboço e um sumário. Por exemplo:

Capítulo 1 - Computadores e Programação
Capítulo 2 – Dados: Tipos de Dados. Constantes e Variáveis.
Capítulo 3 – Dados do Tipo Inteiro.

Tendo definido a estrutura básica, esquematiza-se cada capítulo nos seus mínimos detalhes:

Capítulo 1
- Secção 1.1
  - SubSecção 1.1.1
  - SubSecção 1.1.2
- Secção 1.2
  - SubSecção 1.2.1

Capítulo 2
- Secção 2.1
- Secção 2.2
  - SubSecção 2.2.1

Capítulo 3
...

Em programação, uma boa organização é vital. Um dos métodos para projectar um algoritmo chama-se Top-Down Design.

O processo é bastante semelhante ao processo ilustrado atrás:

1ª Fase

Algoritmo A

2ª Fase

À medida que se pensa mais sobre o problema, poder-se-á chegar à conclusão que, eventualmente o problema possa ser dividido em três partes.

Algoritmo A	Parte 1
	Parte 2
	Parte 3

3ª Fase

Continuando a procurar uma resolução para o problema, poderemos sentir a necessidade de voltar a dividir cada uma das partes em pequenos problemas.

Algoritmo A	Parte 1	Parte 1A
		Parte 1B
		Parte 1C
	Parte 2	Parte 2A
		Parte 2B
		Parte 2C
	Parte 3	Parte 3A

Esta abordagem onde se passa de uma perspectiva global para uma perspectiva particular, pressupões que cada um dos pequenos problemas funcione e sejam analisado e testado, de forma a assegurar que é a solução correcta para o problema proposto.

Desenvolvimento de programas

1. Análise do problema

estudo do problema juntamente com os futuros utilizadores, o resultado desta fase, em termos profissionais, é um documento especificando o que o programa irá fazer, estudos de desenvolvimento, custos, etc. Serve de definição dos objectivos e de garantia do que vai ser feito.

2. Desenvolvimento da solução

É o desenvolvimento do algoritmo ainda de uma forma abstracta, sem ligação à linguagem de programação propriamente dita. Pretende-se reduzir a complexidade, utilizando a metodologia top-down.

3. Programação da solução

Escolha da linguagem a utilizar face às particularidades do problema (por vezes, esta decisão é tomada ainda na fase de análise). Início da programação.

4. Depuração

Aqui detectam-se, localizam-se e corrigem-se erros. Os erros podem ser de duas classes – de natureza sintáctica e de natureza semântica.

> Erros sintácticos: são os erros mais comuns, são também os mais fáceis de localizar e corrigir. Derivam da escrita das instruções, são detectados pelo processador da linguagem (interpretador ou compilador) que produz mensagens de erro indicando qual a instrução que tem o erro.
>
> Erros semânticos: resultam do programa, sintacticamente correcto, ter um significado para o computador totalmente diferente do significado que o programador gostava que ele tivesse.

5. Documentação

Divide-se basicamente em dois tipos: a documentação de utilização, trata-se do manual de utilização e documentação técnica para futuras alterações ou correcções.

6. Manutenção

Consiste na verificação de possibilidades de optimização.